IN THE SPECIFICATION

With reference to published international application no. WO 2004/046881, from which the subject national application originates, please amend/replace the below identified paragraph(s) as indicated, strikeout or double bracketed portions deleted, underlined items added:

➢**Page 1**, please amend th "title" as follows:

SYSTEM AND METHOD OF VISIBLE SURFACE DETERMINATION ~~SYSTEM & METHODOLOGY~~ IN COMPUTER GRAPHICS USING INTERVAL ANALYSIS

➢**Page 1**, in the first ¶ of the "BACKGROUND" (i.e., the last ¶ beginning on that page), at line 9 thereof, please insert --of-- between "and understanding" and "the single most":

Photorealism for computer-generated scenes, that is to say, the production of a computer-generated scene that is indistinguishable from a photograph of the actual scene, as for instance, the elimination of aliasing, remains the "holy grail" for computer graphic artisans. So much so that Jim Blinn has proclaimed: "*Nobody will ever solve the antialiasing problem,*" emphasis original, Jim Blinn, Jim Blinn's Corner Notation, Notation, Notation, 2003, p. 166. In furtherance of a general appreciation and understanding of the single most important obstacle to photorealism, i.e., the antialiasing problem, an overview of heretofore known image synthesizing processing, beginning with the notion of rendering, must be had.

➢**Page 9**, in the last ¶ beginning "Given the above advantages..." at line 3 thereof, replace "(red-blue-green)"

with --(red-green-blue)--:

Given the above advantages, preferred embodiments of the system may be used with any general form of projection. For example, by representing RGB ~~(red-blue-green)~~ (red-green-blue) coloration as three intervals rather than three points, a process for automatic adaptive resolution is possible, i.e., the dimension of a more and more refined rendering interval can be compared to the fidelity of the rendering machine to adaptively stop rendering smaller intervals once an optimum presentation has been obtained. The system is a massively parallel and scalable rendering engine, and therefore useful for distributing across servers in a network grid to optimally create more realistic images from a scene database, or for implementing in a graphics accelerator for improved performance. Moreover, still images or video images may be more efficiently broadcast to remote clients as the interval analysis methods operate directly on the mathematical functions that describe a scene as opposed to a piecewise geometric or tesselated model of the scene, thus providing efficient data compaction for transmission over limited bandwidth connections.

➢**Page 10**, in the only full ¶ thereon beginning "With preferred...," please:

add at line 10 thereof --(i.e., $0(x^2)$)-- after "quadratic convergence";

replace at lines 11 thereof "logarithmic" with --square root--; and,

add at line 12 thereof --(i.e., $0(x^{1/2})$)-- at the end of the ¶, as follows:

With preferred embodiments of the system, an entire scene can be loaded on each computer connected to an output device and synchronously display an image either by sequentially displaying data from each computer, displaying disjoint pieces from each computer, or a combination of both. The system can casually seek edges of objects or transitional areas, i.e., areas with increased levels of information to concentrate the rendering effort. Convergence to the proper visible solution set of a pixel is a deterministic operation which exhibits quadratic convergence (i.e., $0(x^2)$). This is in contrast to point-sampling methods which are probabilistic and exhibit ~~logarithmic~~ square-root convergence (i.e., $0(x^{1/2})$).

➢**Page 12**, in the last full ¶ thereon beginning "FIGS. 14-18..." at line 3 thereof, please insert --and,-- after ";" as follows:

FIGS. 14-18 are schematic depiction of the work flow of the solvers of FIG. 13, namely, SCREEN, PIXEL, COVERAGE, DEPTH, and IMPORTANCE; and,

➢**Page 12**, in the last ¶ thereon beginning "FIGS. 19(a)..." at line 2 thereof, please insert --inversion-- after "interval set," and replace "; and," with --.-- as follows:

FIGS. 19(a)-19(f) are pictorial representations of the operation of an interval set inversion technique for rendering in accordance with the present invention~~; and,~~ .

➢**Page 13**, delete the first full ¶ thereon beginning "FIG. 20..." as follows:

~~FIG. 20 is a depiction of importance filtering in the context of the importance function of FIG. 18.~~

➢**Page 14**, in the first partial ¶:

at line 20 thereof, please insert reference numeral --42-- after "database"; and,

at line 24 thereof, please delete reference numerals "32" and "30", as follows:

This grid technique is the real world analogy to the computer graphic process that forms the basis of modern day digital graphics. FIG. 2 shows the overall process of how a computer graphics system 40 turns a three dimensional digital representation of a scene 42 into multiple two-dimensional digital images 44. Just as the artist uses the cells 24 and 32 (FIG. 1) to divide the representation of an entire scene into several smaller and more manageable components, the digital graphics system 40 divides an image 44 to be displayed into thousands of pixels in order to digitally display two-dimensional representations of three dimensional scenes. A typical computer generated image used by the modern motion picture industry, for example, is formed of a rectangular array of pixels 1,920 wide and 1,080 high. In a conventional digital animation process, for example, a modeler defines geometric models for each of a series of objects in a scene. A graphic artist adds light, color and texture features to

geometric models of each object and an animator then defines a set of motions and dynamics defining how the objects will interact with each other and with light sources in the scene. All of this information is then collected and related in the scene database 42. A network comprised of multiple servers then utilizes the scene database 42 to perform the calculations necessary to color in each of the pixels in each frame 44 that are sequenced together 51 to create the illusion of motion and action of the scene. Unlike the rectangular cells [[32]] on the artist's paper [[30]], a pixel may only be assigned a single color.

➢**Page 21**, in the first partial ¶ thereon,

at line 3 thereof, please replace "(e.g., nxm)" with --(e.g., (n)(m))--;

at lines 6, 7, 19, 21, and 22, replace "nxm" with --(n)(m)--; and,

at line 21, replace "acheive" with --achieve--, as follows:

For example, in a "super sample" operation (i.e., when using lots of rays to compute the color of a pixel), each pixel is point sampled on a regular grid or matrix grid ~~(e.g. nxm)~~ (e.g., (n)(m)). The scene is then rendered nxm times, each time with a different subpixel offset. After each subpixel rendering is completed, it is summed and divided by the sum ~~nxm~~ (n)(m) . The subject approach need not be confined to regular ~~nxm~~ (n)(m) regular grids. A relaxation technique can be used to automatically generate

irregular super-sampling patterns for any sample count. Ultimately, the aforementioned sampling process will create partial antialiased images that are "box filtered" (FIG. 7(a)), however, there is not reason to limit samples to the area of a single pixel. By distributing point samples in the regions surrounding each pixel center, improved antialiasing, but nonetheless deficient and thereby unacceptable, results may be obtained. Geometry can be sampled using a gaussian, or other sample function, in several distinct and known ways, for instance to weight the distribution of point samples, say the ~~nxm~~ (n)(m) box filtered sample of FIG. 7(a), using a gaussian distribution, and thereby ~~acheive~~ achieve a weighted filtering of the ~~nxm~~ (n)(m) matrix as shown in FIG 7(b). As illustrated in FIG. 7(c), when the uniform ~~nxm~~ (n)(m) matrix is abandoned in favor of a gaussian spatial distribution, and there is a homogeneity of weight with regard to the sample points, a so called importance filtering is achieved.

➢**Page 27**, in first partial ¶ thereon, at line 9 thereof, please insert --is-- after "solvers", as follows:

Referring now to FIGS. 10-12, the subject photorealistic image synthesis process is generally shown, with particular emphasis and general specification of the methodology of the subject interval consistency approach, in furtherance of photorealistic image synthesis processing, outlined in FIG. 12, which utilizes Unified Modeling Language (uml). As shown, central to the process are a

plurality of interval consistency solvers. Operatively and essentially linked to the interval consistency solvers is a system input, exemplified in FIG. 10 by a series of generic parametric equations, each function having two or more variables, for example the arguments t, u, and v as shown, and as representatively illustrated in FIG. 11, wherein the "system" is a sphere, the x-y-z functions being parameterized in t, u, v. It is to be understood that the system need not be limited to parametric expressions, which have the greatest utility and are most challenging/problematic, other geometric primitives, or alternate system expressions are similarly contemplated and amenable to the subject methodology and process as is to be gleaned from the discussion to this point. For example, the system can similarly render strictly mathematical formulae selectively input by a user, such as those describing polygons, and bezier surfaces, the later being the singular focus of RenderMan.

➢**Page 29**, in first partial ¶ thereon, at line 8 thereof, please replace "FIG. 12" with --FIG. 13-- as follows:

The solver, more particularly the most preferred components thereof, namely SCREEN, PIXEL, COVERAGE, DEPTH, and IMPORTANCE, are shown in relation to the input (i.e., dim and system), callbacks (i.e., shader), and output (i.e., pixel data and display). The interrelationships between the individual most preferred elements of constituents of the solver, and the general temporal hierarchy

between and among each, as well as their relationships between the callbacks (i.e., the shader) and the output (i.e., the display) are schematically shown in ~~FIG. 12~~ FIG. 13. As will be subsequently discussed in the flow schematics for each of the solvers, and as is appreciated by a reference to the subject figure, hierarchical, iterative sieving progresses, in nested fashion, from the screen solver to the importance solver, with each solver exporting a constraint for which the subsequent solver is to act in consideration thereof. Values from successively embedded solvers are returned as shown, the pixel solver ultimately bundling qualities or character of color, opacity, depth, and coverage, for instance, and "issues" such bundled information package (i.e., a pixel reflecting that scene object subtending same) to the display as shown in furtherance of synthesizing the 2-D array corresponding to the image plane.

➢**Page 30**, in only full ¶ thereon, at line 12 thereof, please replace "(see FIGS. 19(b) et seq. with --(see FIGS. 19(b) seq.)-- as follows:

Chopping of the x-y image plane begins with an initial step analogous to that illustrated in FIG. 19(b). The idea is to parse the x-y image plane to dimensional equate to a pixel. As shown, in the event that initial chopping yields a sub divided x-y area more extensive than a pixel, more chopping is conducted, namely a preferential chopping. More particularly, the nature of the x-y image plane subunit (i.e., a rectangle) is assessed and

characterized as being either "landscape" or "portrait". In the event the subunit is landscape, the x dimension is further split: in the event that the subunit is portrait, then the y dimension is then split. For each iterative step in x or y ~~(see FIGS. 19(b) et seq.~~ (see FIGS. 19(b) et seq.), the arguments t, u, and v, are contracted so as to eliminate values thereof outside the specific or "working" x-y interval (i.e., with each iteration in x and y, it is advantageous to eliminate the t, u, and v values that are not contributing, and thereby potentially contribute to aliasing).

➢**Page 30**, in last partial ¶ thereon, at line 2 thereof, please replace "screen" with --SCREEN-- as follows:

The pixel solver, depicted in FIG. 15, is essentially a liaison between ~~screen~~ SCREEN and the other solvers, acting as a synchronization point and performing a housekeeping function. Preliminarily, PIXEL seeks an answer to the question, is the nature of the x-y interval corresponding to a pixel area, and thereby the t, u, v solutions associated therewith, such that the shader has been invoked (i.e., color and opacity, for example, has been assigned or designated). If the shader has been invoked, by calling upon the coverage solver, no further parsing of the x-y space (e.g., FIGS. 19(b)-19(f)) is required, and the x-y pixel data is sent to the display.

➢**Page 31**, in only full ¶ thereon, at line 7 thereof, please replace "(again, see FIGS. 19(b)-19(f)" with --(again, see FIGS. 19(b)-19(f))-- as follows:

The coverage solver, as detailed in FIG. 16, essentially replicates the iterations of SCREEN, based upon a user defined limit epsilon (eps). COVERAGE, as the name suggests, seeks to delimit, via the retention of contributing t, u, v aspects based upon the user specified chop area "eps," those portions (i.e., areas) of the object surface within the pixel subunit ~~(again, see FIGS. 19(b)-19(f)~~ (again, see FIGS. 19(b)-19(f)). Upon ascertaining the values associated with the x-y space or area, they are added or compiled to provide or define the total coverage of the object surface (i.e., a mapping of the entire x-y space). At this point, analysis, more particularly processing, in x-y space is complete. The next procedural task is a consideration of depth (i.e., assessment of Z(t, u, v) of the parametric system with a fixed or set x and y).

➢**Page 32**, in the first partial ¶ thereon at line 4 thereof (i.e., in the last ¶ at page 31 beginning "The depth solver..."), please replace "(z depth)" with --z depth, i.e., set to an interval at an infinite distance from the viewer-- as follows:

The depth solver, as detailed in FIG. 17, is essentially doing the job of FIG. 17(a). More particularly, DEPTH initially ascertains where in the z dimension, ultimately from the image plane (see FIG. 4 camera space), does the object surface, heretofore defined in x, y, t, u, v aspects, first appear or reside (i.e., in which depth cell), and thereafter step into space, via iterative cells, until the x, y, t, u, v object surface is no longer present in a cell (i.e., cell **X** of FIG. 17(a)). In furtherance thereof, the depth

variable, more accurately, depth function, is initialized for all depth space, namely set to the infinite interval ~~(z depth)~~ (z depth, i.e., set to an interval at an infinite distance from the viewer. Thereafter, t, u, v, contraction begins in the depth field (z0). Subsequently, there is a trivial accept/reject query as to whether there is in fact a depth component of the x-y parameterization, with searching commencing thereafter (z search). For each depth cell, the importance solver (i.e., the t, u, v, chopper wherein a set inversion is executed in t, u, v so as to contract same) is called upon, and it is necessary to next assess if the shader was invoked. If the shader is invoked (i.e., a first visible root is identified), the output of the shader are accumulated into the importance sums and the depth parsing continues in furtherance of accounting for all z components of the x-y object surface, if not, steps, on a cell by cell basis are "walked off." Although the parsing or chopping of z space has been described as a serial or loop type progression, its is certainly amenable to recursive splitting, as the case of the x-y space.

➢**Page 33**, in the first partial ¶ thereon beginning at line 4 thereof, please delete ", and in the present setting, illustrated in FIG. 20" as follows:

The importance solver, as detailed in FIG. 18, when called, essentially completes a set inversion in t, u, v, that is to say, for the smallest x, y, z (i.e., each specific z cell for, or in, which an object surface element x-y resides), t, u, v are to be

narrowed as much or as finely as possible. The function of the importance solver is to fit or optimally match the t, u, v with the x, y, z, in a way that is overreaching, not underreaching. In furtherance thereof, importance filtering is conducted, the notion previously discussed with respect to the filtering of FIGS. 7(a)-(c)~~, and in the present setting, illustrated in FIG. 20~~.